神经自然语言处理方法中的 子词切分(Subword Tokenization)方法综述

刘群
qun.liu@huawei.com
华为诺亚方舟实验室

摘　要

本文对基于神经网络的自然语言处理方法中的子词切分（Subword Tokenization）方法进行了综述。本文首先解释了基于神经网络的自然语言处理方法中面临的由于封闭词表（Closed Vocabulary）所导致的集外词（Out-of-Vocabulary，简称OOV）问题，并介绍了解决这一方法常见的Byte-Pair Encoding（简称BPE）、WordPiece和Unigram三种方法。子词切分之前通常需要做词语切分，而词语切分是跟具体语言高度相关的。SentencePiece提供了一种与语言无关的子词切分方法，可以在输入的句子上直接做子词切分，无需先做词语切分。子词切分有时会存在一些切分不合理和子词表示学习不够充分的问题，本文随后介绍了解决这一问题的子词正则化技术和BPE-Dropout技术。基于字符的子词切分在面对多语言（特别是中日韩等语言）的大字符集时依然存在OOV问题，本文将介绍解决这一问题的一种有效手段：基于UTF-8字节的BPE技术（Byte Level BPE，简称BBPE），及其衍生的基于BBPE的SentencePiece方案。最后本文介绍了ACL2021最佳论文所提出的一种通用的词表最优化技术VOLT。

关键词： 子词切分, Subword Tokenization, BPE, WordPiece, SentencePiece, BBPE

1　问题由来

1.1　封闭词表假设（Colsed Vocabulary Assumption）和集外词（Out-of-Vocabulary）问题

自然语言处理（NLP）所面临的文本的词表（Vocabulary），通常是开放的，并没有任何规定文本中不能出现某些词。但作为一个自然语言处理系统，通常都要假设所处理的词限定在一个有限的词表范围内，这个假设称为封闭词表假设（Closed Vocabulary Assumption）[1]。尤其是对于基于神经网络的自然语言处理方法，由于我们需要把每一个词事先映射到一个词嵌入表示（Embedding），对于预定义词表以外的词（又称为集外词，或者Out-of-Vocabulary Word，简称OOV），我们是无法预知其嵌入表示的，因此也无法处理。这就是自然语言处理系统所面临的集外词问题（简称OOV问题）。

1.2　解决办法之一：集外词替换为UNK

早期基于神经网络的自然语言处理系统，对这一问题通常采用一个简单粗暴的方法：将所

有集外词统一替换为为UNK，意为未知词（Unknown Word）。但这种做法显然不完美，原因有以下几点：

- 很多集外词虽然在训练数据中没有出现过，但其意义并不是完全不可知的，相反，相当多的集外词的意义是可以从已知词推导出来的，把它们都替换成UNK，就无法利用这些词的语义信息。比如说：
  - 屈折变化词：对某些词，词表中可能没有收录其所有的变化形式（如英语名词复数、形容词比较级、最高级、动词现在分词、过去分词等），这些没有收录的变化形式就会被识别为集外词，如词集中可能有cascade，但没有cascaded；
  - 复合词：比如hard-working、thirteen-years等；
  - 数词：数词是不可能全收的，没有收录的数词都会成为集外词。
- 在自然语言理解（NLU）任务中，出现UNK的问题还不太大，对系统性能的影响不会很严重，但在自然语言生成（NLG）任务中，如机器翻译、对话生成等任务中，生成的句子中如果出现很多UNK是很严重的问题，用户会无法接受。简单删掉这些UNK又会使得句子不通顺。

1.3 解决办法之二：基于字符的模型

基于字符的模型把模型建立在字符的基础上，词的表示（word embedding）通过字符的表示（character embedding）经过一个神经网络计算得到。基于字符的模型可以较好地解决OOV问题，但也会带来序列长度的大大增加，而模型处理长序列的难度比处理短序列大得多。通常为了达到类似的性能，基于字符的模型需要要比基于词的模型要复杂得多。

而对于中日韩（CJK）语言这样的大字符集语言来说，基于字符的模型需要的词表依然太大。Unicode 1.0.1（1992）定义的CJK统一字符集已经有20902个字符，而最新的Unicode 14.0（2021）已经包含字符总数144,697个，把这么多的字符全部收入模型，是不合理的。而且绝大部分字符都是罕用字符，使用概率极低，全部收入词表会大大增加模型参数。不仅如此，由于每次生成一个字符需要在整个词表上做softmax操作，如此大的词表，而且词表中大部分都是罕见字符，也会导致模型推理效率大大降低。

2 BPE和WordPiece：子词（subword）级别的词表构造和子词切分方法

2.1 子词（subword）级别的切分：解决集外词问题的有效手段

如前所述，词级别的词表无法解决集外词OOV问题，而字符级别的词表又存在模型复杂度高和效率低的问题。为此，学者们提出了子词（Subword）级别的词表构造和切分方法[2-3]，可以较好地解决这一问题。子词级切分方案的基本思想是：

- 词表只收录固定的子词集合；
- 任何一个词，如果不在词表中，就需要被切分成子词序列（subword token sequence）。

这种方法的优点有：

- 彻底解决了集外词OOV问题，任何一个单词都可以切分成子词序列。最坏情况下，所有子词都匹配不上，就会切分成字符序列；

- 大部分常用词都可以作为独立子词收入词表，这样的话原始的基于词的模型无需做任何改动，序列长度增加很少，模型复杂度和效率都几乎没有增加。

接下来的问题是：（1）如何确定基本的子词词表？（子词词表构造问题）（2）如果确定了子词词表，如何将句子切分成子词序列？（子词切分问题）

对于大部分西方语言，一种很容易想到的方法，是将词语切分成词根和词缀，词根和词缀就是子词。但这种办法有以下问题：（1）不是所有的词都可以切分成词根和词缀的，比如外来词和数词等；（2）词语切分成词根词缀的过程通常被称为形态分析（morphological analysis）或者词法分析（lexical analysis），而这个过程是跟语言高度相关的，无法做到语言通用性。

目前，最流行的两种构造子词词表和子词切分的方法分别是BPE和WordPiece，还有一种使用较少的Unigram方法，下面我们分别介绍。

2.2 Byte-Pair Encoding (BPE)

BPE最早是作为一种压缩算法提出来的，[2]首次提出将BPE算法用于子词级切分，以解决神经机器翻译的集外词问题，效果非常明显。很快，这一方法也被用于其他解决其他问题，并成为基于神经网络的NLP模型中最为流行的子词切分方法。

采用BPE算法做词例切分，首先需要使用语料库构造一个子词词表（Vocabulary）。下面我以表1为例将构造子词词表的过程介绍如下：

1. 预处理文本，得到一部词典（Dictionary），词典中包含文本中出现的所有单词，以及该单词出现的频率；
2. 将词典中所有的单词切分成基本字符的序列，字符和字符之间加上一个空格；
3. 初始化子词词表（Vocabulary），词表只包含所有的基本字符（如字母、数字、标点、汉字等）构成的子词subword；
4. 重复以下过程，每一步将子词词表中的两个子词合并（merge）得到一个新的子词加到词表末尾，直到子词词表Vocabulary的长度达到预设值：
   (a) 在词典中选择出现频率最高的子词对儿（subword pair）；
   (b) 将上述子词对儿合并成一个新的子词，并到子词词表Vocabulary末尾；
   (c) 在词典中出现的所有上述子词对儿都合并成新加入子词（删掉原来两个子词之间的空格）；
5. 返回词表。

基于BPE的子词切分（subword tokenization）非常简单：对于一个给定的文本，首先把单词之间的空白都替换成</w>，然后把所有单词都切分成字符（任何两个字符直接都加一个空格），然后根据BPE的子词词表从前往后，对于每个合并（merge）形成的子词，在文本中反复执行其对应的合并操作，直到所有合并操作都执行完毕。

2.3 WordPiece

Google最早在[4]中介绍了WordPiece的思想，但并没有命名这个算法。在论文[3]中Google首次提到在神经机器翻译系统中使用了WordPiece做文本的子词切分。后来Google在BERT中再次

freq	step 0	step 1	step 2	step 3	step 4	step 5
5	l o w </w>	l o w </w>	l o w </w>	l o w </w>	lo w </w>	low </w>
2	l o w e r </w>	l o w e r </w>	l o w e r </w>	l o w e r </w>	lo w e r </w>	low e r </w>
6	n e w e s t </w>	n e w es t </w>	n e w est </w>	n e w est</w>	n e w est</w>	n e w est</w>
3	w i d e s t </w>	w i d es t </w>	w i d est </w>	w i d est</w>	w i d est</w>	w i d est</w>

freq	7	7	13	16	17	2	6	9	9	3	3	9	9	9	7	7
step 0	l	o	w	</w>	e	r	n	s	t	i	d					
step 1	l	o	w	</w>	e	r	n	s	t	i	d	e+s				
step 2	l	o	w	</w>	e	r	n	s	t	i	d	e+s	es+t			
step 3	l	o	w	</w>	e	r	n	s	t	i	d	e+s	es+t	est+</w>		
step 4	l	o	w	</w>	e	r	n	s	t	i	d	e+s	es+t	est+</w>	l+o	
step 5	l	o	w	</w>	e	r	n	s	t	i	d	e+s	es+t	est+</w>	l+o	lo+w

表 1: BPE子词切分过程示例，上表为词典Dictionary，下表为子词词表Vocabulary。表中</w>是一个特殊字符，用于表示一个单词的结尾。step 0是初始化的词典和子词词表，其他每一个step将出现频率最高subword pair合并成一个新的subword，并加在子词词表末尾。

使用了WordPiece，使得WordPiece影响大增。但Google并没有开源WordPiece的词表构造算法，也没有公开源代码。好在BERT的代码中还是包括了WordPiece的词例切分工具（Tokenizer），所以如果研究者们不想构造自己的子词词表，还是可以直接使用BERT提供的词表和词例切分工具做研究的。后来Huggingface的工具包中也提供了一个WordPiece模型的训练工具，但他们并没有真正使用[4]中的算法，实际上还是使用BPE方法来构造了一个子词词表，只是把生成的词表改成了BERT提供的WordPiece数据格式。

WordPiece的子词词表构造算法跟BPE非常相似，也是一个从基本字符集开始逐渐合并的过程，每次合并都在词表中增加一个新的子词。区别在于，BPE在合并两个子词的时候，只需要计算两个子词连续出现的概率，而WordPiece则复杂得多。WordPiece首先需要根据给定的子词词表构造一个n-gram语言模型，然后用这个语言模型可以计算整个训练数据出现的似然率（likelihood）。合并的时候，考虑所有的子词对儿（subword pair），根据该子词对儿合并后得到的新的语言模型计算整个训练数据的似然率，选择导致整个训练数据似然率最高的那个子词对儿进行合并。这个过程比较复杂，计算代价很高。Google估计是设计了某种优化算法才有可能实现高效的词表构建，但这个算法没有公开。

推理时，WordPiece子词切分采用从左到右的最大匹配方法，也就是一种贪心方法，非常简单。实际上BPE的子词切分也可以采用这种方法，效果跟自底向上合并（merge）的方法上应该没有太大区别。

2.4 Unigram

Google还提出了另外一种词表构造方法，叫做Unigram方法[5]。Unigram跟WordPiece一样，也采用了语言模型来决定词表中是否收录某个子词。WordPiece原始论文[4]并没有明确说明所使用的语言模型，而Unigram方法使用的是一元语法语言模型，这也是这种方法取名为Unigram的原因。不过跟BPE和WordPiece不同的是，Unigram构造词表的方式不是先构造字符级别词表，再

通过合并操作逐渐往词表中增加新的子词，相反，Unigram先构造一个大的子词词表（比如收录语料库中所有词），然后再逐渐删除词表中的子词（把罕见的子词切分成两个更常见的子词），直到词表长度达到预定的值。Unigram词表构造方法在一个开源的工具SentencePiece（后面会提到）中提供了具体实现，但运行效率比较低，实际上使用这种方法的人很少。

推理时，Unigram的子词切分也采用从左到右的最大匹配方法。

3 SentencePiece: 与语言无关的子词切分方法

仔细研究上面介绍的子词切分方法，我们会看到，我们在做子词切分（subword tokenization）之前，还需要先做一次词语切分（word tokenization），以得到一个初始的词典（也就是前面BPE词表构造方法中的Dictionary）。但词语切分也并不是一件简单的事情，比如：

- 在英语中，标点符号和字母、数字之间通常是没有空格的，尤其是英语的句号（.）和数词中的小数点、缩写词中的点号完全相同，需要专门的算法来排除歧义；
- 在中日韩等语言中，字符（主要是汉字）之间是没有空格的，需要专门引入外部的词语切分工具进行词语切分；
- 每一种语言都有各自的词语切分问题，词语切分无法做到语言无关。

SentencePiece[6]就是为了解决这一问题提出的子词切分方法。SentencePiece的基本思想是，对于输入的文本无需先做词语切分，直接在输入文本串的基础上构造子词词表和进行子词切分。直观地理解，可以认为输入的每个句子（甚至段落）就是一个单词，然后再构造词表进行子词切分。在这个方法中，空格不作为词语之间的分隔符，也就是说，空格与其他任何字符同等对待。按照这种方式得到的子词词表，子词中出现空格，或者子词中同时出现字母、数字或者标点符号都是很常见的。论文[6]的实验表明，采用SentencePiece的方法，无需先做词语切分，可以取得跟先做词语切分方法可比的效果。

SentencePiece也可以用于处理中文，无需事先做中文分词就可以直接使用。也有些人先对文本做中文分词，然后再调用SentencePiece，也是完全可以的。不过既然先做了词语切分，后面调用SentencePiece和直接调用BPE的效果是一样的。另外要特别注意：推理的时候也需要先用相同的中文分词工具做词语切分，然后再调用SentencePiece做子词切分，否则训练和推理子词切分过程不一致，会导致效果很差。

SentencePiece和WordPiece这两个名词有点容易造成混淆，让人以为SentencePiece是跟WordPiece一样的一种子词切分方法。实际上这是两个不同层次的概念。WordPiece和BPE、Unigram是类似的技术，都是用于构造子词词表和做子词切分的方法。SentencePiece方法只是强调子词切分之前无需做词语切分，直接从句子开始构造子词词表和进行子词切分，但并没有规定子词词表构造和子词切分的算法。所以即使采用SentencePiece方法，还是需要选择WordPiece、BPE或者Unigram来做子词词表构造和子词切分的。

另外要注意的是，SentencePiece不仅仅是一种子词切分方法，也是一套开源的工具。这套工具提供了两个子词词表构造的工具：BPE和Unigram。

4 子词正则化方法和BPE-Dropout方法：改进子词切分不合理和子词表示训练不充分的问题

做过中文分词的人都知道，词语切分是有很多歧义的，子词切分也不例外。如果深入观察BPE或者WordPiece产生的子词切分结果，会发现很多不合理的切分。

BPE切分方法还会导致一些子词训练不充分的问题。比如一个串ABC有可能切分成A/BC或者AB/C，其中A/BC是合理切分，AB/C是不合理切分。但由于语料库数据分布的问题，很可能导致语料库中AB/C出现的次数大大高于A/BC，这样BC虽然是一个高频子词，但在训练数据中出现的次数却非常少，这就会导致BC这个子词的表示训练得非常不充分。以表1为例，这个例子中，w+e实际上是一个高频的子词序列，但我们学到的子词词表甚至没有学到这个子词。

[5]提出了一种子词正则化的方法来解决子词切分歧义的问题。其思想还是引入一个语言模型（如Unigram），对于多种切分结果，根据语言模型的概率随机取样其中一种切分结果，这样会使得切分结果比较合理。但这种做法的代价也是很大的，因为需要引入一个语言模型，会导致训练过程中的计算量大大增加。

[7]提出一种更加简单的BPE-Dropout技术，用于改善BPE方法的子词表示训练不充分的问题。其思想非常简单，就是在语言模型训练阶段做子词切分（Subword Tokenization）的时候，以一定的概率随机跳过一些子词合并（merge）操作，这样通过增加子词切分的随机性，就可以使一些原来训练不充分的低频子词出现概率大大增加，从而学到更好的子词表示。

另外要注意的是，无论是子词正则化还是BPE-Dropout，这种在切分过程中引入随机性的做法，通常都是只在模型训练阶段使用，以确保得到更好的模型。在语言模型应用（推理）阶段，为了保持一致性和提高效率，通常还是采用确定性的子词切分方法。

5 Byte-level BPE (BBPE): 解决大字符集语言的OOV问题

我们在第1.3节中提到，子词切分的方法，对于中日韩这样的大字符集语言来说，还是有问题的：（1）字符太多，甚至可能超过预定的词表大小，导致一些字符无法收入子词词表，仍然会有OOV问题；（2）即使字符集足够大，把所有字符都收入子词词表，这会导致词表中收入大量罕见字，词表空间利用率大大降低，而在解码的时候，需要在词表的所有子词空间上计算softmax，大量的罕见字出现概率极低，这样也导致解码的时间效率大大降低。

论文[8]给出了一个mBERT子词词表（WordPiece词表）中每个子词在一个训练语料库中出现的频率分布图（图1）。从图中可以看出，这个频率分布在超过词表长度60%以后就快速下降，在超过83%以后直接降到0。说明BERT的词表使用是非常低效的。即使这样mBERT仍然存在OOV问题；我们在使用中发现，著名作家"章诒和"的"诒"如果写成繁体"詒"，就无法被mBERT识别。

Byte-level BPE（简称BBPE）为这一问题的解决提供了有效的办法。 BBPE的基本思想是：把所有的字符表示成UTF-8格式，把这个UTF-8字符串当成一个字节组成的序列，然后以一个字节为最小的组成单位，来构造BPE子词词表和进行子词切分。这样的话，BPE最基本的组成单位最多只有256个，不存在大量罕见字符占用子词词表空间的问题。

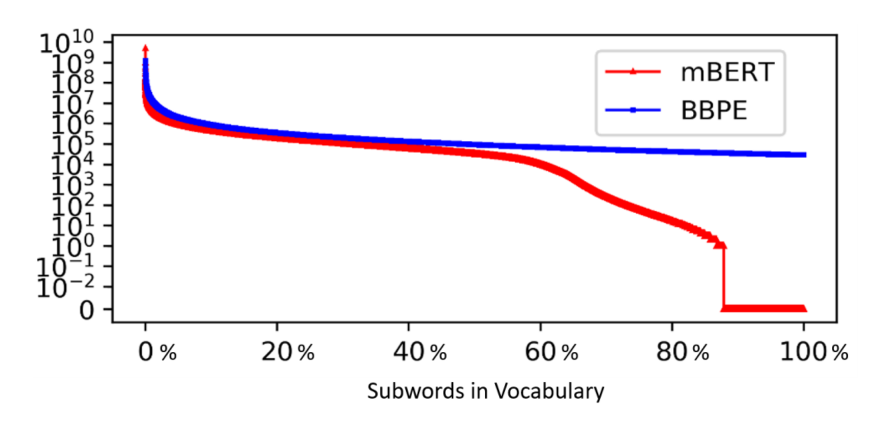


图 1: mBERT词表和BBPE词表中的子词在训练数据中出现的概率分布

根据UTF-8的编码规则，一个字符可以表示成1-4个字节，长度不等。这样会带来一个问题，一些罕见字符会被切分成两个或者多个子词，理论上有可能导致生成的时候会出现非法字符（非法子词序列），但实际上这种情况几乎不会出现。因为在训练数据中几乎不会出现非法子词序列，这种概率分布是会被神经语言模型学习到的，模型生成的时候也会遵从这种概率分布，因此生成非法子词序列的可能性虽然理论上存在，但概率极低，实际上几乎不会发生。我们在使用BBPE训练GPT模型的时候，也从来没有遇到过生成非法字符的情况。

BBPE方法首先在GPT-2模型[9]中被采用，后来RoBERTa模型[10]也采用了BBPE方法做子词切分，随后的GPT-3模型[11]也延续了这种方法。论文[12]和[8]分别对BBPE在神经机器翻译和预训练语言模型中的使用效果进行了深入分析。Huggingface工具包也提供了BBPE模型的具体实现。我们（华为诺亚方舟实验室）在我们的开源预训练语言模型代码库[1]中也提供了BBPE的完整代码。另外考虑到已有的BBPE实现都没有对SentencePiece的支持，而SentencePiece使用的人还是很多的，我们的代码库中还提供了基于BBPE的SentencePiece的实现方案。

6 VOLT：子词词表最优化方法

在构建子词词表的时候，有一个问题就是：子词词表规模要选择多大比较合适？理论上，词表越大，包含的信息越多，模型效果当然更好。但模型越大，占用的空间就越多，推理时间也会相应增加，代价也会越来越高。那么在词表大小和系统性能之间是否存在一个合理的平衡点呢？论文[13]提出了一种VOLT模型，对这一问题给出了一个明确的答案。该论文获得了ACL 2021的最佳长论文奖。

由于下游任务的性能衡量并没有统一的方法，该论文采用词表的信息熵来粗略代表下游任务的性能：词表越大，词表对应的信息熵就越低，相应的，下游任务的效果就好越好，反之亦然。这样问题就转化成如何找到一个最优的子词词表，可以在词表的信息熵和词表的大小之间寻找一个合理的平衡点。

为了寻找这个最优的平衡点，该论文引入了边际收益的概念：将词表的增大理解为投入，词表信息熵的降低理解为收益。该论文定义了词表边际收益（MUV）这个衡量指标，就是每增

[1]https://github.com/huawei-noah/Pretrained-Language-Model

加固定的词表规模带来的信息熵降低的幅度。在词表很小的时候，加大词表带来的边际收益是很高的，随着词表越来越大，这种边际收益会越来越低。

为了获得一个MUV最大的词表，该论文把词表搜索问题转成了一种最优运输问题，该问题可以在多项式时间内用动态规划方法求解。在52个方向的机器翻译实验表明，采用该方法可以将词表规模降低到基线系统的30%，但机器翻译的性能总体持平甚至略有提高。

7 总结

本文主要综述了基于神经网络的自然语言处理方法中采用的子词切分方法。我们首先介绍了子词切分问题的来由：封闭词表假设所带来的集外词（OOV）问题，以及为什么要采用子词切分来解决这一问题。其次我们介绍了构造子词词表和子词切分的常用方法，包括BPE、WordPiece和Unigram。然后我们介绍了子词切分技术所面临的一些相关问题及其解决办法，包括：（1）子词切分之前需要先进行词语切分的问题，以及避免该问题可以采用的SentencePiece方法；（2）切分歧义问题和子词表示训练不充分问题，以及解决这一问题的子词规范化方法和BPE-dropout方法；（3）大字符集问题，以及解决这一问题的BBPE方法；（4）词表最优化问题，以及解决这一问题的VOLT方法。

子词切分是神经NLP方法的最基本的技术之一，全面了解这一技术有利于更好地理解基于神经网络的NLP模型，并设计更好的NLP系统。

参考文献

[1] BOTHA J A, DYER C, BLUNSOM P. Bayesian language modelling of german compounds[C/OL]//Proceedings of COLING 2012. Mumbai, India: The COLING 2012 Organizing Committee, 2012: 341-356[2021-11-14]. https://aclanthology.org/C12-1022.

[2] SENNRICH R, HADDOW B, BIRCH A. Neural machine translation of rare words with subword units[J]. arXiv preprint arXiv:1508.07909, 2015.

[3] WU Y, SCHUSTER M, CHEN Z, et al. Google's neural machine translation system: Bridging the gap between human and machine translation[J]. arXiv preprint arXiv:1609.08144, 2016.

[4] SCHUSTER M, NAKAJIMA K. Japanese and korean voice search[C]//2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2012: 5149-5152.

[5] KUDO T. Subword regularization: improving neural network translation models with multiple subword candidates[C/OL]//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, 2018: 66-75[2021-08-30]. https://aclanthology.org/P18-1007. DOI: 10.18653/v1/P18-1007.

[6] KUDO T, RICHARDSON J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing[J]. arXiv preprint arXiv:1808.06226, 2018.

[7] PROVILKOV I, EMELIANENKO D, VOITA E. Bpe-dropout: Simple and effective subword regularization[J]. arXiv preprint arXiv:1910.13267, 2019.

[8] WEI J, LIU Q, GUO Y, et al. Training multilingual pre-trained language model with byte-level subwords[J]. arXiv preprint arXiv:2101.09469, 2021.

[9] RADFORD A, WU J, CHILD R, et al. Language Models are Unsupervised Multitask Learners[J]. 24.

[10] LIU Y, OTT M, GOYAL N, et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach[EB/OL]. (2019-07-26) [2021-11-27]. http://arxiv.org/abs/1907.11692.

[11] BROWN T B, MANN B, RYDER N, et al. Language Models are Few-Shot Learners[EB/OL]. (2020-07-22)[2020-12-26]. http://arxiv.org/abs/2005.14165.

[12] WANG C, CHO K, GU J. Neural machine translation with byte-level subwords[C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 34. 2020: 9154-9160.

[13] XU J, ZHOU H, GAN C, et al. Vocabulary learning via optimal transport for neural machine translation[C]//Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2021: 7361-7373.